محمد امانلو — تکلیف اول هوش مصنوعی — 810100084 — استاد فدایی

---

① Uninformed search

frontier = {A, D, B, E, H, C, F, G, F, I}
(cost shown for each node: A 0, D 14, B 32, E 39, H 40, C 39, F 50, G 44, F 49, I 46)

explored = {A, D, B, C, E, H, G, I}

total cost = 46

frontier = {A}, explored = {}

frontier = {B_{32}, D_{14}}, explored = {A}

frontier = {B_{32}, E_{39}}, explored = {A, D}

frontier = {E_{39}, H_{40}, C_{39}}, explored = {A, D, B}

frontier = {E_{39}, H_{40}, F_{50}, G_{44}}, explored = {A, D, B, C}

frontier = {H_{40}, F_{49}, G_{44}}, explored = {A, D, B, C, E}

frontier = {F_{49}, I_{46}, G_{44}}, explored = {A, D, B, C, E, H}

frontier = {F_{49}, I_{46}}, explored = {A, D, B, C, E, H, G}

به هدف با cost 46 رسیدیم.

• توضیح مرحله به مرحله:

① frontier ← A

② frontier ← D_{14}, B_{32} / explored ← A

③ frontier ← E_{39} / explored ← D
مسیر: A → D

④ frontier ← H_{40}, C_{39} / explored ← B
مسیر: A → B

⑤ frontier ← G_{44}, F_{50} / explored ← C
مسیر: A → B → C

⑥ frontier ← F_{49} / explored ← E
delete F_{50}
مسیر: A → D → E

⑦ frontier ← I_{46} / explored ← H
مسیر: A → B → H

⑧ frontier ← × / explored ← G
مسیر: A → B → C → G

⑨ ✓ / explored ← I
A → B → H → I

مسیر نهایی: A → B → H → I

هیوریستیک پیشنهادی این است که فاصله‌ی هر عنصر با جایگاه درست خودش را از لحاظ میزان قدر مطلق ارتفاع بدست آورده و روی تمامی node ها جمع ببندیم و تقسیم بر ۲ کنیم.

* علت تقسیم بر ۲:

در صورتی که شکل مقابل را در نظر بگیریم: a در جایگاه درست a

و طبق مطالب باشد با یکبار جابجایی a و b در شرایط واقعی (Cost) $\frac{1}{2}$ تا هزینه

درحالی که ما هیوریستیک ما قبل از تقسیم بر ۲ مقدار ۲ را نشان داده است پس باید تقسیم بر ۲ کنیم (بدترین حالت این است)

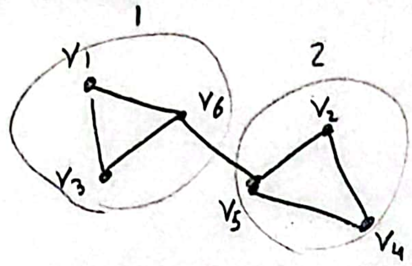
* در واقع در جابجایی هر ۲ عنصر دلخواه ۳ حالت وجود دارد:

۱) جابجایی a و b (Cost=۱) باعث شود a یکی دور شود و b یکی نزدیک شود (یکی بین یک ارتفاع) یعنی h_0 ما - چون نزدیک شدن a یکی کم شود - چون دور شدن b یکی اضافه شده است یعنی h_0 تغییری نمی‌کند (هیوریستیک ثابت است) $\Rightarrow$ مقدار تغییر h بین دو state از cost کمتر است

۲) جابجایی a و b (Cost=۱) باعث شود یا هر ۲ از جایگاه اصلی دور شوند یا هر ۲ به جایگاه اصلی نزدیک شوند در این حالت مجموع فاصله‌ی ارتفاع‌ها به اندازه ۲ تا زیاد می‌شود که پس از تقسیم بر ۲ همان عدد ۱ که مساوی Cost است حاصل می‌شود. در نتیجه تفاوت Cost های بین ۲ state با جابجایی این ۲ عنصر ۱ است و تفاوت h های آن‌ها نیز یک است. در حالت دیگر نیز تفاوت h های دو state صفر و Cost بین آن‌ها ۱ است که در این حالت نیز تفاوت بین h های ۲ state از Cost بین ۲ state کمتر است. یعنی consistent heuristic است، چون در صورتی که یک هیوریستیک consistent باشد حتماً admissible نیز هست پس هیوریستیک ما هم admissible و هم consistant است.

$$\rightarrow h = \frac{\sum |height(i) - height(i_t)|}{2}$$

الف) هر کروموزوم شامل n ژن است که هر ژن نماینده‌ی یک node است. و شامل شماره‌ی گروه آن node است که به صورت Random مشخص می‌شود. شماره‌ی گروه‌ها می‌توانند هر عدد رندومی بین ۱ تا n (n تعداد node ها است)



ب)

v_1	v_2	v_3	v_4	v_5	v_6
1	2	1	2	2	1

ج) هر ژن را با احتمال P_m با یک شماره گروه جدید در بازه‌ی ۱ تا n جایگزین می‌کنیم.

د) هر ژن از هر کروموزوم را با احتمال P_c با همان عنصر (ژن) از کروموزوم دیگر جابجا می‌کنیم.

هـ) یکی از fitness function های مناسب برای هر کروموزوم استفاده از شاخص sillouhete است. این شاخص برای یک دسته‌بندی (clustring) معین ثبات خوشه‌بندی را نمایش می‌دهد.

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j)$$

که C_i یعنی خوشه‌ای که ما node شماره i را به آن اختصاص داده‌ایم و $d(i, j)$ در واقع فاصله‌ی node های i و j است. درصورتی که بین i و j یالی باشد مقدار یک و در غیر این صورت مقدار صفر می‌گیرد. در نهایت ما می‌توانیم $a(i)$ را به عنوان اینکه چه مقدار i به خوشه خود اختصاص دارد در نظر بگیریم. حال باید میانگین عدم شباهت نقطه i را به خوشه‌هایی مانند C_k به عنوان میانگین فاصله از تمامی نقاط داخل C_k تعریف می‌کنیم.

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j)$$

آن خوشه‌ای که در min خارج شده است یعنی خوشه همسایه i است (بیشترین شباهت یا حداقل مقدار ایل) در سایر خوشه‌ها را دارد. حال مقدار سیلوئت را از روش روبرو حساب می‌کنیم

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))} \quad |C_i| > 1$$

در صورتی که $a(i) = b(i)$ باشد $s(i) = 0$ است در این حالت،

$$s(i) = \begin{cases} 1 - a(i)/b(i) & \text{if } a(i) < b(i) \\ 0 & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1 & \text{if } a(i) > b(i) \end{cases}$$

از تعریف بالا مشخص می‌شود که $-1 < s(i) < 1$

حال با تعریف Fitness Function می‌توانیم، با ازای هر کروموزوم مجموع $s(i)$های ژن‌های آن را Fitness Func تعریف می‌کنیم.

چون $s(i)$ نزدیک به ۱ یعنی $a(i) \gg b(i)$ است و از آنجایی که $a(i)$ یک معیار برای خلوتی عدم شباهت تا با خوشه خودش است و بزرگ بودن $b(i)$ یعنی تا با خوشه‌های همسایه تطابق خوبی نداشته است. پس $s(i)$ نزدیک به ۱ یعنی خوشه‌بندی مناسب تا است.

در مقابل اگر $s(i)$ به معنی یک نزدیک باشد با همان منطق بالا متوجه می‌شویم که آمد تا در خوشه‌ی درستی بود (خوشه همسایه) تطابق بیشتری داشت.

پس زیاد شدن مجموع $s(i)$ها یعنی یک دسته‌بندی خوب و کم بودن آن یعنی یک دسته‌بندی بد که این می‌تواند یک شاخص Fitness مناسب برای مسئله باشد.

MDP ○

سئوال اول)

• در اینجا ما ۲ پالیسی برای policy داریم،

	S_0	S_1	S_2	S_3
π_1	a_1	a_0	a_0	a_0
π_2	a_2	a_0	a_0	a_0

$$V^*(S_0) = \max_a 0 + \gamma \sum_{s'} T(s,a,s') V^*(s') = \gamma \max\{V^*(s_1), V^*(s_2)\}$$

$$V^*(S_1) = \max_a 0 + \gamma \sum_{s'} T(s,a,s') V^*(s') = \gamma[(1-p)V^*(s_1) + pV^*(s_3)]$$

$$V^*(S_2) = \max_a 1 + \gamma \sum_{s'} T(s,a,s') V^*(s') = 1 + \gamma[(1-q)V^*(s_0) + qV^*(s_3)]$$

$$V^*(S_3) = \max_a 10 + \gamma \sum_{s'} T(s,a,s') V^*(s') = 10 + \gamma V^*(s_0)$$

• توجه کنید (بعدی نمیدانیم $p<0$)

$$\pi^*(S_0) = \arg\max_a \gamma \sum_{s'} T(s,a,s') V^*(s') \longrightarrow \gamma \neq 0 \text{ برای}$$

درصورت ۰ بودن هیچ π، یکتایی بدست نمی‌آید.

پس درصورتی که $V^*(S_2) > V^*(S_1)$ باشد، $\pi^*(S_0) = a_2$

حال p را بررسی می‌کنیم. درصورتی که $p<0$ باشد چون $V^*(S_1)$ را صفر می‌شود پس درحالت $(p<0)$

$$V^*(S_2) = 1 + \gamma[(1-q)V^*(S_0) + qV^*(S_3)] \geq 1$$

چون دیدیم،

$$V^*(S_2) > V^*(S_1)$$

پس،

و قبلاً دیدیم که چون $T(S,a_2,s')$ برابر ۱ است. پس

$$\pi^*(S_0) = a_2$$

می‌شود حالت فوق $\pi^*(S_0)$ بین $V^*(S_2)$ و $V^*(S_1)$، ما $V^*(S_2)$ را انتخاب می‌کنیم.

پس به ازای $p<0$ تضمین می‌شود که $\pi^*(S_0) = a_2$ است حرکه به ازای a_1 قطعاً جریمه reward دریافت می‌کنیم. ✓

• این مسئله دقیقاً برعکس مسئله قبل است.

جواب منفی است؟ خیر نمی‌توانیم.

$$v^*(s_2) = 1 + \gamma[(1-q)v^*(s_0) + q v^*(s_3)] \geq 1$$

چرا که

پس اگر بخواهیم در هر صورت در state s_0، action a_0 را انتخاب کنیم پس باید الزاماً

$$v^*(s_1) > v^*(s_2)$$

باشد. می‌دانیم $v^*(s_2) \geq 1$ است. چون

$$v^*(s_1) = \gamma[(1-p)v^*(s_1) + p v^*(s_3)] = \frac{\gamma p v^*(s_3)}{1-\gamma(1-p)}$$

حال چون ما هیچ محدودیتی برای γ نداریم، ما همواره می‌توانیم یک γ پیدا کنیم که $v^*(s_1) < 1$ باشد و به این دلیل شرط لازم برای آنکه $v^*(s_1) > v^*(s_2)$ باشد برقرار نیست (یعنی می‌توان یک γ یافت که این شرط را نقض کند) پس هیچ مقداری برای q نداریم که با آن الزاماً $\pi^*(s_0) = a_0$ باشد.

•

$\gamma = 0.9 \quad p = q = 0.25 \quad \epsilon = 10^{-3}$ ✓

جواب نهایی این مسئله،

$v^*(s_0) = 14.184 \quad \pi^*(s_0) = a_1$

$v^*(s_1) = 15.760 \quad \pi^*(s_1) = a_0$

$v^*(s_2) = 15.696 \quad \pi^*(s_2) = a_0$

$v^*(s_3) = 22.766 \quad \pi^*(s_3) = a_0$

$$v^*(s_0) = \max_a \{ [1 \times (0 + \gamma v^*(s_1))], [1 \times (0 + \gamma v^*(s_2))] \} = 0.9$$

$$v^*(s_2) = \max_a \{ [0.25 \times (1 + 0.9 \times 10) + 0.75 \times (1 + \gamma v^*(s_1))] = 3.8575$$

$$v^*(s_1) = \max_a \{ 0.25 \times (0 + 9) + 0.75 \times (0 + 0) \} = 2.25$$

$$v^*(s_3) = \max_a \{ 1 \times (10 + \gamma 0.9) \} = 10.81$$

iteration 1

همین ترتیب را ادامه می‌دهیم.

یک iteration قبل این هست که از 0 مقادیر 0,1,0,10 را روی s_0, s_1, s_2, s_3 می‌گذاریم

$$V^*(s) = \max_a \sum_{s'} T(s,a,s')(R + \gamma V^*(s'))$$

$V^*(S_0) = \max_a \{ [1 \times (0 + \gamma \times 2.25)], [1 \times (0 + \gamma V^*(S_2)] \} = 3.47175$

0.9×3.8575

$V^*(S_1) = \max_a T(S,a,S')[R(S,a,S') + \gamma V^*(S')] = 3.951$

$V^*(S_2) =$ ——— $= 7.461$

$V^*(S_3) =$ ——— $= 14.678$

به همین ترتیب ادامه می‌دهیم تا زمانی که میزان خطای هر اتریشن با در واقع تفاوت ولیو با ولیوی اتریشن قبلی کمتر از ε که همان 10^{-3} است شود. بین کانورج شده است و مقادیر V به‌دست آمده مقادیر درست هستند. همچنین آن a که با بالت V^* ها شود نیز optimal است.

که در این سئوال:

$V^*(S_0) = 14.184$ $\quad \pi^*(S_0) = a_1$

$V^*(S_1) = 15.76$ $\quad \pi^*(S_1) = a_0$

$V^*(S_2) = 15.696$ $\quad \pi^*(S_2) = a_0$ $\quad$ بعد از 53 تا iteration

$V^*(S_3) = 22.766$ $\quad \pi^*(S_3) = a_0$

سه توضیح به‌دست آوردن 10, 1 :

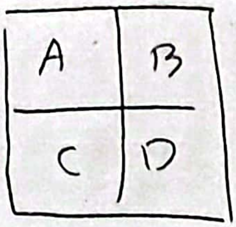
$V^*(S_0) = 0$, $V^*(S_1) = 0$

$V^*(S_2) = 0.25 \times (1 + 0.9 \times 0) + 0.75 \times (1 + 0.9 \times 0) = 1$

$V^*(S_3) = 0.25 \times (10 + 0.9 \times 0) + 0.75 \times (10 \times 0.9 \times 0) = 10$

سؤال دوم

۱) فضای حالت: چون از حضور یا عدم حضور مسافران مطلع نیستیم پس فضای حالت دو تا تاکسی و شرایط آن استفاده است.



تاکسی در A, B, C, D است → 4
×
تاکسی مسافر دارد / ندارد → 2

8 حالت

البته جایگاه مسافران بیرونی کمک است اهمیتی ندارد چرا که این ما احتمالی برای وجود مسافر داریم. اطلاع از وجود آن پس در هر جایگاه مثلاً دستور سوار کردن مسافر را اعلام می‌کنیم. درصورتی که مسافر منتظر بود

۲) ما ۷ تا action داریم، ۱) منتظر ماندن ۲) سوار کردن ۳) پیاده کردن ۴) راست ۵) چپ ۶) بالا ۷) پایین

۳)
* درصورتی که تاکسی مسافر داشته باشد و دستور سوار کردن مسافر را بزند [illegible] به خود آن استیت برگشت می‌شود (تغییر نمی‌کند)
* درحالتی که تاکسی مسافری ندارد و دستور سوار کردن را می‌زند به احتمال p مسافر سوار می‌شود و به احتمال 1-p مسافری سوار نمی‌شود و در همان state می‌ماند
* درحالتی که تاکسی مسافری ندارد و دستور پیاده کردن را می‌زند در همان state می‌مانیم
* درحالتی که تاکسی مسافری دارد و دستور پیاده کردن را می‌زند مسافر پیاده می‌شود و به state که تاکسی در همان جایگاه و بدون مسافر است می‌رویم
* درصورت انجام حرکت از دستورات چپ، راست، بالا، پایین نیز طبق آن به state جدید می‌رویم
* در دستور منتظر ماندن هم در همان state می‌مانیم.

در تمامی حالات - احتمالاتی نیز ممکن است با انجام یک دستور آن دستور اجرا نشود و دستور دیگری اجرا شود به جز جاهایی که مسافر نداریم و می‌خواهیم آن را پیاده کنیم و جاهایی که مسافر داریم و می‌خواهیم سوار کنیم.

۴- ما برای هر عمل مانند سوار کردن، پیاده کردن و ... یک Reward در نظر می‌گیریم.

در واقع هر بار جریمه باقی ماندن در محیط (انتظار) و جریمه زندگی برای هر خانه، جایزه‌ی سوار کردن مسافر در جایگاه درست، جایزه پیاده کردن در جای درست و جریمه‌ی پیاده کردن در جایگاه اشتباه را باید بعنوان Value برای state در نظر گرفت. حال مثلاً اگر تاکسی نابلد جریمه‌ی زندگی در محیط صفر می‌شود (یا در صورتی که مسافری نداشته باشیم جریمه‌ی پیاده کردن صفر می‌شود) - بنابراین حالات مختلف مجموع این جرایم و جوایز را محاسبه می‌کند و بعنوان Reward - agent داده می‌شود.

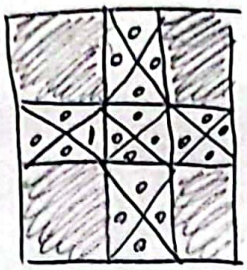
در ضمن برای رسیدن به هدف صدور سئوال باید جریمه‌ی زندگی از جریمه‌ی منتظر ماندن بیشتر و جریمه‌ی پیاده کردن اشتباه بیشتر و همچنین جایزه‌ی پیاده کردن درست از باقی بیشتر باشد.

در نهایت با محاسبه‌ی تابع جریمه و جایزه‌ها برای هر action یک q-value تعریف کرده تا با الگوریتم Convergence پیش می‌رویم و بیشترین q-value را انتخاب می‌کنیم بعنوان π^* در نظر می‌گیریم.

RL 0

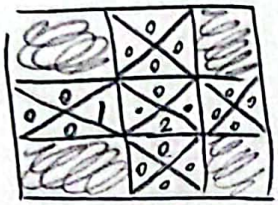
$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha[R(s,a,s') + \gamma \max_{a'} Q(s',a')]$$

($\alpha = 0.5$, $\gamma = 1$)



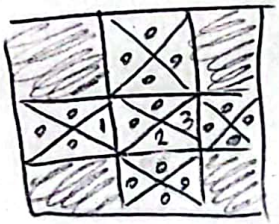
(B, East, C, 2) -1

$(1-0.5)\times 0 + 0.5[2 + 1\times 0] = 1$



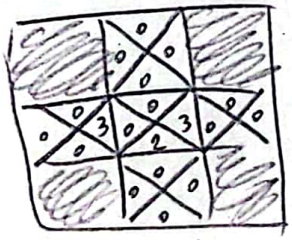
(C, South, E, 4) -2

$(1-0.5)\times 0 + 0.5(4 + 1\times 0) = 2$



(C, East, A, 6) -3

$(1-0.5)\times 0 + 0.5\times(6 + 1\times 0) = 3$



(B, East, C, 2) -4

$(1-0.5)\times 1 + 0.5\times(2 + 1\times 3) = 3$

→ q-table ✓

ب) در حالتی که ما از روش مذکور استفاده کنیم در واقع policy درست را داریم و تنها باید Q ها را پیدا کنیم.
در اینجا با احتمال ε-1 روی policy درست iterate میکنیم و درصدی که با احتمال ε نیز یک
action نادرست داشته باشیم در تعداد iteration های بالا باز Value های درست را استخراج میکنیم. و چون
در حال استفاده از ε-greedy هستیم پس در صورت کاهش ε نیز در مراحل بالاتر انحراف در Value درست را کمتر
داریم.

مقایسه: ۱) Performance : با این دانستن جواب درست این روش بهتر اجرا میشود.

۲) در الگوریتم Q-learning محدودی هرگونه خطا در تقریب هنگام بروزرسانی Q-table قابل
اصلاح است اما در این روش این احتمال با قوه کمتر است اما نیز خطا نیز بسیار کمتر است

۳) همچنان مقایسه کاسم میتوان گفت که، به دلیل اینکه هر بار در حال بهتر کردن همان Q-value
که تا آن در policy هم نیز داریم هستیم این مقادیر سریعتر و دقیق‌تر بروزرسانی میشوند

به طور کلی میتوان نتیجه‌گیری کرد گفت وقتی از policy همیشه پیروی میکنیم عملکرد الگوریتم بهبودی
بهتر از عملکرد Q-learning عادی است. زیرا احتمال انتخاب عمل اشتباه تا ε است.